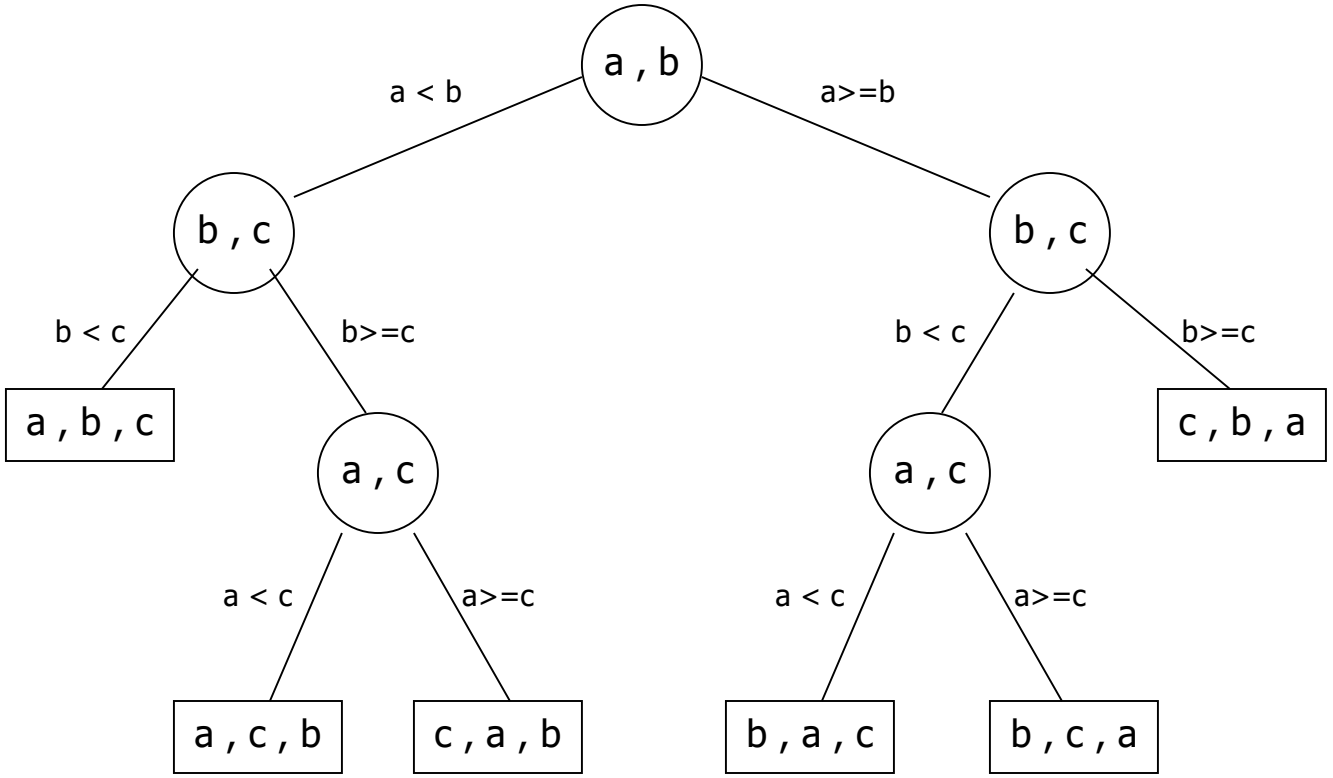
Esercizi C

Esercizio

- Scrivere un programma in linguaggio C che, letti tre numeri interi *a, b, c* dallo standard input, stampi a terminale la sequenza dei tre numeri in ordine crescente.
- Esempio: a = 10, b = 7, c = 9 deve dare in uscita 7 9 10.

- Si può sicuramente condurre un'analisi per casi e, in base a quello individuato, scrivere le tre variabili in ordine non decrescente.



```
#include <stdio.h>
 int main( ) {
      int a,b,c;
      printf("Inserisci il numero a: ");
      scanf("%d",&a);
      printf("Inserisci il numero b: ");
      scanf("%d",&b);
      printf("Inserisci il numero c: ");
      scanf("%d",&c);
```

```
 if (a < b) {
   if (b < c) { printf("\nIn ordine: %d, %d, %d",a,b,c);  }
   else {
     if (a < c) { printf("\nIn ordine: %d, %d, %d",a,c,b); }
     else { printf("\n In ordine: %d, %d, %d",c,a,b); }
   }
 }
 else {
   if (c < b) { printf("\nIn ordine: %d, %d, %d",c,b,a); }
   else {
     if (a < c) { printf("\nIn ordine: %d, %d, %d",b,a,c); }
     else { printf("\nIn ordine: %d, %d, %d",b,c,a); }
   }
 }
 return 0;
}
```

Oppure

```
if (a < b && b < c) {
   printf("\nIn ordine: %d, %d, %d",a,b,c);
}
else if (a < b && b >= c && a < c ) {
   printf("\nIn ordine: %d, %d, %d",a,c,b);
}
else if…

return 0;
}
```

- Una seconda strategia di soluzione facilmente generalizzabile consiste nello scambiare ordinatamente le tre variabili finché i loro contenuti non risultino ordinati.
- A tal fine quindi sarà necessaria sicuramente almeno un'altra variabile intera strumentale allo scambio tra due variabili.
- Stesura informale dell'algoritmo:
  - Leggi i tre numeri *a, b, c*
  - confronta i valori di *a* e *b*, se non sono ordinati si effettua lo scambio
  - confronta i valori di *a* e *c*, se non sono ordinati si effettua lo scambio
  - confronta i valori di *b* e *c*, se non sono ordinati si effettua lo scambio
  - Stampa a video delle tre variabili *a, b, c*

```
#include <stdio.h>
int main( ) {
  int a,b,c,temp;
  printf("Inserisci il numero a: ");  scanf("%d",&a);
  printf("Inserisci il numero b: ");  scanf("%d",&b);
  printf("Inserisci il numero c: ");  scanf("%d",&c);

  if (a > b) {   temp = a;   a = b;   b = temp;  }
  if (a > c) {   temp = a;   a = c;   c = temp;  }
  if (b > c) {   temp = b;   b = c;   c = temp;  }

  printf("In ordine: %d, %d, %d",a,b,c);
  return 0;
}
```

Scambiare il contenuto di due variabili è come scambiare il liquido di due bicchieri, serve un terzo bicchiere di «appoggio»

- La seconda soluzione pur essendo più semplice della soluzione 1, dal punto di vista della struttura e quindi anche della leggibilità, è meno efficiente: effettua sempre tre confronti diversamente dalla prima soluzione che ne effettua due in due casi su sei.
- A ben vedere quindi la soluzione 1 risulta essere la soluzione *ottima*: non è possibile trovarne una che effettui un numero inferiore di confronti.

Esercizio MAXSEQ

- **Scrivere un programma che dato un numero N>0 di valori da inserire da tastiera, stampi a video il massimo della sequenza inserita e la posizione in cui tale valore è stato inserito.**
- **Supponiamo, per semplicità, che non ci siano duplicati**
- **Esempio: N=5 sequenza: 3, 2, 9, 5, 1**

  Max=9 Pos=3

Algoritmo

dichiaro le variabili:N,elemento,max,posizione del max,contatore
leggo N
se N>0
leggo primo valore
assumo che sia il massimo e quindi la sua posizione è quella del max
ripeto per tutti i numeri rimanenti
lettura dell'elemento
incremento il contatore delle posizioni
se l'elemento è > di max
dico che il max è quello appena letto
la posizione del massimo è quella attuale
stampa max e posizione del max
altrimenti
stampa che il valore inserito non va bene */

```c
#include <stdio.h>
int main()  {
    int N, elemento, max, posmax, i=1;
    printf("Inserire numero di valori:\n"); scanf("%d", &N);
    if(N>0) {
        printf("Inserire il primo valore:"); scanf("%d", &elemento);
        max=elemento;
        posmax=i;
        while(i<N) {
          printf("Inserire un altro valore:"); scanf("%d", &elemento);
          i=i+1;
          if(elemento>max) { max=elemento; posmax=i; }
        }
        printf("Il massimo e': %d\n", max);
        printf("La posizione del massimo e': %d\n", posmax);
    }
    else
        printf("Il valore di N non e' accettabile\n");
    return 0;
}
```

Esercizio

- Si scriva un programma in linguaggio C che letto un numero intero positivo dallo standard input, visualizzi a terminale il quadrato del numero stesso facendo uso soltanto di operazioni di somma.
- Si osservi che il quadrato di ogni numero intero positivo N può essere costruito sommando tra loro i primi N numeri dispari.
- Esempio: N = 5; $N^2 = 1 + 3 + 5 + 7 + 9 = 25$.

```
#include <stdio.h> /* inclusione della libreria standard    */
int main( ) {
  int i, N, S=0;
  do { /* Finché il numero inserito N non è positivo ripetere */
     printf("\n Inserisci un numero positivo N: ");
     scanf("%d",&N);
  } while (N < =0 );
  i=0;
  while(i < N) {
      S = S + (i+i+1);
      i++;
  }
  printf(“Il quadrato del numero inserito e': %d \n",S);
  return 0;
}
```

Varianti

```
i=1;
while(i <= N) {
    S = S + (i+i-1);
    i++;
}
```

oppure

```
i=1;
while(i < N+N) {
    if(i%2!=0)
        S = S + (i);
    i++;
}
```

oppure

```
i=1;
while(i < N+N) {
    S = S + i;
    i=i+2;
}
```

Esercizio

- Si scriva un programma in linguaggio C che letto un numero intero positivo dallo standard input, visualizzi a terminale il cubo del numero stesso facendo uso soltanto di operazioni di somma.

```
#include <stdio.h> /* inclusione della libreria standard     */
int main( ) {
  int i, N, S=0, cubo=0;
  do { /* Finché il numero inserito N non è positivo ripetere */
     printf("\n Inserisci un numero positivo N: ");
     scanf("%d",&N);
  } while (N < =0 );
  i=0;
  while(i < N) {
      S = S + (i+i+1);
      i++;
  }
  //Qui S è uguale al quadrato di N
  i=0;
  while(i < N) {
      cubo=cubo+S;
      i++;
  }
  printf("Il cubo del numero inserito e': %d \n",cubo);
  return 0;
}
```

Esercizio

- Scrivere i primi 30 elementi di una serie così definita: i primi tre elementi valgono 1,i successivi (i>=4) valgono la somma degli elementi i-1 e i-3
- 1 1 1 2 3 4 6 9 13 19 28 41 60 …

```
#include <stdio.h>
int main() {
    int quanti=30,i,elem1=1,elem2=1,elem3=1,elem;
    printf("1 1 1”);
    i=4;
    while(i<=quanti) {
        elem=elem1+elem3;
        printf(" %d",elem);
        elem3=elem2;
        elem2=elem1;
        elem1=elem;
        i++;
    }
    return 0;
}
```

Esercizio

- Si scriva un programma che legge una sequenza di interi positivi (la sequenza termina quando viene inserito il valore -1), conta il numero complessivo dei numeri che sono multipli di 3, di 5 oppure di 7 compresi nella sequenza e stampa questo valore. Per esempio, nel caso la sequenza in ingresso fosse "4 8 12 15 14 8", il programma dovrebbe stampare il valore 3.

```
# include <stdio.h>
int main() {
    int val, contatore=0;
    printf(“Inserisci una serie di interi (-1 per terminare):\n");
    do {
        scanf("%d",&val);
        if (val>-1 && (val%3==0 || val%5==0 || val%7==0))
                contatore++;
    } while(val!=-1);
    printf("Il numero di multipli di 3 o 5 o 7 è %d",contatore);
    return 0;
}
```

Esercizio

- Scrivere un programma C, in grado di acquisire in ingresso dall'utente un valore intero num e una sequenza di interi che termina con uno 0 (zero). Il programma deve stampare a video il numero di valori pari nella sequenza che sono divisori di num.
- 0 viene considerato come valore sentinella.

```
# include <stdio.h>
int main() {
    int num, valore, cont=0;
    printf("Inserisci un numero intero \n");
    scanf("%d",&num);
    printf("Inserisci una serie di interi (0 per terminare):\n");
    do {
        scanf("%d",&valore);
        if (valore!=0 && (valore%2)==0 && (num%valore)==0)
                cont++;
    } while(valore!=0);
    printf("I valori pari divisori di %d sono %d",num,cont);
    return 0;
}
```

```
# include <stdio.h>
int main() {
    int num, valore, cont=0;
    printf("Inserisci un numero intero \n");
    scanf("%d",&num);
    printf("Inserisci una serie di interi (0 per terminare):\n");
    while(1) { // soluzione orribile
        scanf("%d",&valore);
        if(valore==0)
                break;
        if ( valore%2==0 && num%valore==0)
                cont++;
    }
    printf("I valori pari divisori di %d sono %d",num,cont);
    return 0;
}
```

```
# include <stdio.h>
int main() {
    int num, valore, cont=0;
    printf("Inserisci un numero intero \n");
    scanf("%d",&num);
    printf(“Inserisci una serie di interi (0 per terminare):\n");
    while(valore) { // soluzione un po’ meno orribile
        scanf("%d",&valore);
        if (valore!=0 &&  valore%2==0 && num%valore==0)
                cont++;
    }
    printf(“I valori pari divisori di %d sono %d“,num,cont);
    return 0;
}
```

Esercizio

- Si scriva un programma che legge una sequenza di caratteri (la sequenza termina quando viene inserito il carattere "#"),conta il numero complessivo di vocali minuscole ("a", "e", "i", "o", "u") comprese nella sequenza e stampa questo valore.
- Per esempio, nel caso la sequenza in ingresso fosse

  defghi123jklmaAAa002#

  il programma dovrebbe stampare il valore 4.

```
# include <stdio.h>
void main() {
    char car;
    int contatore=0;
    printf(“Inserire una serie di caratteri(# per finire)\n");
    do {
        scanf("%c",&car);
        if(car=='a’||car=='e’||car=='i’||car==‘o’||car=='u')
                contatore++;
    } while(car!='#');
    printf("Il numero delle vocali e' %d",contatore);
}
```

Esercizio – variante tutte lettere

- Si scriva un programma che legge una sequenza di caratteri (la sequenza termina quando viene inserito il carattere "#"),conta il numero complessivo di lettere minuscole comprese nella sequenza e stampa questo valore.
- Per esempio, nel caso la sequenza in ingresso fosse

  defghi123jklmaAAa002#

  il programma dovrebbe stampare il valore 12.

```
# include <stdio.h>
void main() {
    char car;
    int contatore=0;
    printf("Inserire una serie di caratteri(# per finire)\n");
    do {
        scanf("%c",&car);
        if( car>='a' && car<='z' )
                contatore++;
    } while(car!='#');
    printf("Il numero delle lettere minuscole e' %d",contatore);
}
```

```
# include <stdio.h>
void main() {
    char car;
    int contatore=0;
    printf("Inserire una serie di caratteri(# per finire)\n");
    do {
        scanf("%c",&car);
        if('a'<=car<='z' ) // NOOOOOOOOOOOOOO
                contatore++;
    } while(car!='#');
    printf("Il numero delle lettere minuscole e' %d",contatore);
}
```

```
# include <stdio.h>
void main() {
    char car;
    int contatore=0;
    printf("Inserire una serie di caratteri(# per finire)\n");
    do {
        scanf("%c",&car);
        if( car>='a' && <='z' ) //NO, ogni espressione deve
                                //    essere COMPLETA
            contatore++;
    } while(car!='#');
    printf("Il numero delle lettere minuscole e' %d",contatore);
}
```

Esercizio Divisori Primi

- Scrivere un programma C, completo delle opportune dichiarazioni di variabili, in grado di acquisire in ingresso dall'utente un valore intero positivo num. Il programma deve stampare a video tutti i fattori primi di num.

```
# include <stdio.h>
int main() {
	int num, fatt, cont;
	printf("\nInserisci un numero intero \n");
	scanf("%d",&num);
	printf("\n i fattori primi di %d sono:",num);
	for (fatt=num;fatt>1;fatt--)  { //ipotesi: se num è primo considero num
                                    //          divisore primo di se stesso
		if(num%fatt==0) { /*se fatt è divisore di num cerca se ha divisori*/
			cont=fatt-1; // tutti i numeri son divisibili per se stessi, quindi fatt-1
			while((cont > 1) && (fatt%cont != 0))
				cont--;
			/* se sono arrivato a 1 vuol dire che non ho trovato divisori
			quindi stampo fatt perché primo */
			if (cont == 1)
				printf(" %d ",fatt);
		}
	}
	return 0;
}
```

Esercizio Scomposizione Fattori Primi

- Scrivere un programma C, completo delle opportune dichiarazioni di variabili, in grado di acquisire in ingresso dall'utente un valore intero num. Il programma deve stampare a video l’intera scomposizione in fattori primi di num.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int num, div;
    do {
        printf("Scrivi il numero che vuoi scomporre (positivo)\n");
        scanf("%d", &num);
    } while (num<=0);
    printf("La sua scomposizione in fattori primi e':\n");
    for(div=num-1;div>0;div--){
        if (num%div==0){
            printf("  %d  ", num/div);
            num=div;
        }
    }
    system("PAUSE");
    return 0;
}
```

```
# include <stdio.h>
int main() {
        int num, i=2, cont;
        printf("\nInserisci un numero intero \n");
        scanf("%d",&num);
        printf("\n i fattori primi di %d sono:",num);
        if(num==1) /*trattiamo il caso num=1 a parte*/
                printf(“1“);
        while(num!=1)  {
                cont=0;
                while(num%i==0) {
                        cont++;
                        num=num/i;
                }
                printf(“%d^%d “,i,cont);
                i++;
        }
        return 0;
}
```

Esercizio

- Si definisce *Triangolare* un numero costituito dalla somma dei primi N numeri interi positivi per un certo N.
- Ad esempio: per Q = 10 si ha Q =1+2+3+4, da cui N = 4.
- Scrivere un programma C che stabilisca se un numero intero positivo Q, letto dallo standard input, è un numero triangolare o meno, utilizzando soltanto operazioni tra numeri interi. In caso affermativo stampare a video il numero inserito e il massimo degli addendi che lo compongono.

```
#include <stdio.h>    /* inclusione della libreria standard     */
int main( ) {
 int i=0, Q, S;
 do {
   printf("\n Inserisci un numero positivo Q: "); scanf("%d",&Q);
 } while (Q <= 0);
 S = Q;        /* copia del valore del dato in ingresso       */
 while (S > 0) {
   i = i + 1;
   S = S - i;
 }
 if (S == 0) {
  /*i contiene qui il valore del massimo addendo componente il numero*/
   printf("\n %d = alla somma dei primi %d numeri positivi!",Q,i);
 } else {
  printf("\n Il numero %d non e' un numero triangolare! \n",Q);
 }
 return 0;
}
```

Esercizio

- Dato un numero positivo Q, scrivere la sua rappresentazione in binario naturale, indicando anche il minimo numero di bit utilizzato.
- Il programma dovrà esibire un comportamento come nell'esempio seguente:

  Input: 19 in decimale

  Output: con 5 bit = 10011 in binario.

- **Idea di soluzione:** se $Q = q_{n-1}2^{n-1}+q_{n-2}2^{n-2}+\ldots+q_1 2+q_0$, posso confrontare Q con le successive potenze di 2: $1,2,2^2,2^3,\ldots$ finché risulta $Q >= 2^j$. Quando si verificherà la condizione per cui $Q < 2^n$, l'esponente di tale potenza sarà proprio il numero di bit necessario a rappresentare Q.
- **Si osservi che**
  - $q_{n-1} = 1$ perché per costruzione $Q >= 2^{n-1}$, rappresentato su n bit
  - $q_{n-2} = 1$ sse $(Q - q_{n-1}2^{n-1}) >= 2^{n-2}$ altrimenti $q_{n-2} = 0$
  - $q_{n-3} = 1$ sse $(Q - q_{n-1}2^{n-1} - q_{n-2}2^{n-2}) >= 2^{n-3}$ altrimenti $q_{n-3} = 0$
  - ……
  - $q_0 = 1$ sse $(Q-q_{n-1}2^{n-1} - q_{n-2}2^{n-2}-\ldots- q_1 2^1) >= 1$ altrimenti $q_0 = 0$

  allora…

- Leggi il numero intero *Q* da convertire
- Inizializza un contatore *n* al valore *0*
- Inizializza un accumulatore *d*, per le potenze di *2*, al valore *1*
- Finché (il numero da convertire è >= *d*) esegui
  - incrementa di uno il contatore n
  - moltiplica per *2* la variabile *d*
- Stampa a video il valore della variabile *n*
- Inizializza un contatore *i* con *n-1*
- Finché (*i >=0*) esegui
  - dividi per *2* la variabile *d*
  - se (*Q >= d*) allora
    - stampa a video un carattere “1”
    - assegna a *Q* il valore *Q-d*
  - altrimenti stampa a video il carattere “0”
  - decrementa di *1* il contatore *i*

```
#include <stdio.h> /* inclusione della libreria standard       */
int main( ) {
 int i,d=1,n=0,Q;
 do{printf("Inserire un numero positivo:");scanf("%d",&Q);}while(Q<=0);
 while (Q >= d)  { n = n+1;  d = d*2; }
 /* d è la più piccola potenza di 2 maggiore di Q, n l'esponente della
     potenza di 2 in d e il numero di bit minimo per rappresentare Q. */
 printf("\n%d in decimale, con %d bit = ",Q,n);
 /* esponente più significativo della rappresentazione binaria di Q. */
 i = n-1;
 while(i >= 0) {
   d = d / 2;
   if (Q>=d) {printf("1"); Q=Q-d; }
   else { printf("0"); }
   i--;
  }
 return 0;
}
```

Esercizio

- Si scriva un programma in linguaggio C che riceve dallo standard input due caratteri alfabetici, li converte in maiuscolo e stampa a video ordinatamente tutti i caratteri dell'alfabeto fra essi compresi, estremi inclusi.
- Esempio: dati 'g' e 'M‘ stampa a video la sequenza: GHIJKLM.

- Ricordando che nella tabella ASCII si hanno le seguenti corrispondenze tra i caratteri alfanumerici e le loro codifiche:
  - '0' - '9' : 48 - 57
  - 'A' - 'Z' : 65 - 90
  - 'a' - 'z' : 97 – 122
- seguendo quanto richiesto nella traccia del problema è possibile formulare la soluzione come segue:
  - Leggere due caratteri alfabetici in due variabili x, y
  - Converte in maiuscolo i caratteri inseriti
  - Se ($x > y$) allora scambia il valore delle due variabili
  - Inizializza un contatore i con il valore del codice ASCII del carattere x
  - Finché il contatore è minore o uguale al codice ASCII del carattere y (stampa il carattere con codice ASCII uguale al valore del contatore)

```
#include <stdio.h> /* inclusione della libreria standard     */
int main( ) {
 char x,y,t;
 do {
   printf("\n Inserisci il carattere x: "); scanf("%c",&x);
 } while (!(((x >= 'a')&&(x <= 'z'))||( (x >= 'A')&&(x <= 'Z'))));
 do  {
   printf("\n Inserisci il carattere y: "); scanf("%c",&y);
 } while (!(((y >= 'a')&&(y <= 'z'))||((y >= 'A')&&(y <= 'Z'))));

 /* converte i caratteri nel corrispondente maiuscolo */
 if ((x >= 'a')&&(x <= 'z')) x = x - 'a‘ + 'A';
 if ((y >= 'a')&&(y <= 'z')) y = y - 'a‘ + 'A';

 if (x > y) { t = x;  x = y;  y = t; } /* ordina x,y */

 printf("\n La sequenza di caratteri richiesta e': ");
 while (x<=y) { printf("%c",x); x = x+1; }
 return 0;
}
```

Esercizio

- Si scriva un programma in linguaggio C che risolva il problema seguente. Leggere una sequenza di caratteri alfanumerici dallo standard input terminata dal carattere ‘#’; quindi stampare sullo standard output la media dei numeri interi corrispondenti ai caratteri numerici inseriti.
- Esempio:

  input: ‘A’ ‘b’ ‘C’ ‘1’ ‘F’ ‘4’ ‘G’ ‘T’ ‘6’ ‘Y’ ‘6’ ‘3’ ‘s’

  output: media = 4

```
#include <stdio.h> /* inclusione della libreria standard    */
int main( ) {
 char  c;   int accumulator=0, count=0;  float mean;
 do {
  printf("\n Inserisci un carattere: ");  scanf("%c",&c);
  if ((c >= ‘0')&&(c <= '9')) {
     accumulator = accumulator + ((int)c – ‘0’);
     count++;
  }
 } while (c!=’#’);
 if (count > 0) {
    mean = ((float)accumulator) / ((float)count);
    printf("\n media = %5.2f ",mean);
 } else {
     printf("\n non ci sono caratteri numerici!");
 }
 return 0;
}
```

Esercizio

- Dato un carattere in ingresso, trasformarlo in un altro carattere, che si trova OFFSET posizioni più in là nell'alfabeto
- L'alfabeto considerato è:
  - ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
- L'alfabeto è ciclico: dopo la 'z' c'è la 'A'
- Per esempio, con OFFSET = 4
  - il carattere 'a' diventa 'e'
  - la lettera 'X' diventa 'b'
  - la lettera 'x' diventa 'B'
- La complessità del programma sta tutta nel fatto che nella codifica ASCII le sequenze 'A'..'Z' e 'a'..'z' (viene prima la sequenza di caratteri maiuscoli) non sono consecutive, ma c'è di mezzo un altro insieme di caratteri, per cui occorre spezzare il programma in 2 if.

```
#include <stdio.h>
#define OFFSET 4
int main() {
    char curr, new;
    printf("Inserisci il carattere alfabetico da convertire: ");
    scanf("%c", &curr);
    if (curr >= 'A' && curr <= 'Z') {
        new = curr + OFFSET;
        if (new > 'Z') {
            new = new – 'Z' + 'a' -1;
        }
        printf("Il nuovo carattere dopo la conversione e': %c\n", new);
    } else if (curr >= 'a' && curr <= 'z') {
        new = curr + OFFSET;
        if (new > 'z') {
            new = 'A' + (new – 'z'–1);
        }
        printf("Il nuovo carattere dopo la conversione e': %c\n", new);
    } else { printf("Non e' stato immesso un carattere valido\n"); }
    return 0;
}
```

Esercizio (tdeB 22/11/2007)

- Si caratterizzi sinteticamente la serie di numeri stampati da numeri (in funzione di n)

```
int main() {
  int i, j, k, n;
  scanf(“%d”,&n);
  for( i = 2; i <= n; i++ ) {
    k = 0;
    for( j = 2; i%j != 0; j++ )
      k++;

    if( j != i )
      k--;
    else
      printf("%d ", i);
  }
  printf("\n");

   return 0;
}
```

- La funzione stampa (in ordine crescente) tutti i *numeri **primi*** compresi tra 2 e n (estremi inclusi).
- Si osserva innanzitutto che la variabile k è sempre solo scritta e mai letta, e non può avere alcun effetto sulla stampa dei valori di i. Si può quindi riscrivere il codice come segue:

```
int main() {
  int i, j, n;
  scanf("%d",&n);
  for( i = 2; i <= n; i++ ) {       // per ogni i da 2 a n incluso
   for( j = 2; i%j != 0; j++ )    // incrementa j finché non divide i
     ;                        // (j è il min. div. non banale di i)
   if( j == i )                 // se tale divisore è i stesso
     printf("%d ", i);          // (e cioè se i è primo) stampalo
  }
  printf("\n");
  return 0;
}
```

- Si nota quindi che il ciclo esterno considera tutti i numeri i compresi tra 2 e n. Di ognuno di tali il i il ciclo più interno cerca il minimo divisore j che non sia banale (j riparte ogni volta da 2 e cresce fino al primo valore che divide i).
- Se tale valore è i stesso, significa che i è primo, e solo in questo caso viene stampato.
- Diversamente, si passa a considerare il valore i successivo.
- Se n è minore di 2 la funzione non stampa nessun numero.

Esercizio (tdeB 24-11-2005)

- Si dica quale proprietà di due numeri interi è verificata dal programma seguente, argomentando brevemente la risposta.

```
int main() {
        int a,b,c;
        scanf("%d",&a);
        scanf("%d",&b);
        if ( a < 0 )
                a = -a;
        if ( b < 0 )
                b = -b;
        if ( a < b ) {
                c = a;    a = b;    b = c;
        }
        while ( a > b )
                a -= b;

        printf ("%d", a == b );

        return 0;
}
```

```
int main() {
    int a,b,c;
    scanf("%d",&a);
    scanf("%d",&b);
    if ( a < 0 )      /* a è riassegnato al suo valore assoluto */
          a = -a;
    if ( b < 0 )      /* b è riassegnato al suo valore assoluto */
          b = -b;
    if ( a < b ) {    /* se b è maggiore di a, scambia i valori di a e b */
          c = a;
          a = b;
          b = c;
    }  /* a questo punto, quindi, è certamente a >= b */
    while ( a > b )    /* sottrae b ad a finché non diventa a <= b */
          a -= b;
    printf ("%d", a == b ); /*restituisce 1 se sono diventati uguali,
                                                    0 se diversi*/
}
```

Soluzione:

Se a e b sono uguali oppure entrambi diversi da 0, la funzione verifica **se uno dei due è multiplo dell'altro**, provando a ridurre tramite sottrazioni successive il modulo del maggiore (in valore assoluto) al modulo dell'altro.

La funzione NON termina se uno dei due è pari a zero e l'altro no (in tal caso, infatti, entra nel ciclo while e non ne esce più).

Dato che 0 è multiplo di ogni numero relativo, per implementare in modo ineccepibile la verifica della proprietà il programmatore avrebbe potuto/dovuto iniziare la funzione con

```
if ( a==0 || b==0  )
        return 1;
```

Giustificazione analitica:

Dopo le prime tre istruzioni if, a e b rappresentano i valori assoluti di a e b, in ordine descrescente (a >= b). Continuando a decrementare a del valore di b, solo due situazioni possono verificarsi all'uscita del ciclo (che avviene a patto che non sia $a \neq 0$ e $b = 0$):

a diventa UGUALE a b, e quindi i parametri iniziali erano uno multiplo dell'altro. Si noti che se a e b sono uguali da subito tutte le condizioni precedenti alla printf sono false, e il valore stampato è correttamente 1.

a diventa MINORE di b, e a e b non erano uno multiplo dell'altro. In particolare, in questo caso a rappresenta il resto della divisione intera tra il maggiore (in modulo) tra i parametri e il minore (in modulo) tra i parametri.

Esercizio

- Scrivere un programma che stampa i primi 20 numeri altamente composti.
  - Un numero altamente composto è tale che qualunque numero minore di esso ha meno divisori. I primi numeri altamente composti sono 1, 2, 4, 6, 12, 24, 36, 48, 60, 120, 180, 240, 360, 720, 840, 1260, 1680, 2520, 5040, 7560, 10080

Esercizio

- Su una scacchiera 8x8 sono posizionati due pezzi: il Re bianco e la Regina nera.
- Si scriva un programma in linguaggio C che, acquisite le posizioni del Re e della Regina, determini se la Regina è in posizione tale da poter mangiare il Re. Le posizioni dei due pezzi sono identificate da mediante la riga e la colonna su cui si trovano, espresse come numeri interi tra 1 e 8.

Numeri mancanti, perfetti, abbondanti

Scrivere un programma che legge un intero positivo n da stdin e verifica se n è un numero mancante, perfetto o abbondante. Chiamiamo $\sigma(n)$ la somma di tutti i divisori propri di n (1 incluso, n escluso). Un numero n si dice *perfetto* se $n = \sigma(n)$, mancante se $n > \sigma(n)$, abbondante se $n < \sigma(n)$. Esempio:

```
10: MANCANTE
12: ABBONDANTE
28: PERFETTO
```

Scomposizione in somma di quadrati

Scrivere un programma che legge un intero positivo n da stdin e verifica se n può essere scomposto nella somma di **due** quadrati (verifica cioè se $\exists$ a, b $\in$ N | $a^2+b^2=n$). Se sì, stampare a video la scomposizione. Esempi:

```
2 ==> 2 = 1 + 1 = 1^2 + 1^2
28 ==>  NON SCOMPONIBILE
145 ==> 146 = 25 + 121 = 5^2 + 11^2
```

Varianti e aggiunte (per la meditazione domestica)

a. Mostrare, quando ve ne è più di una, tutte le diverse scomposizioni dello stesso numero (ad esempio 50 ha due scomposizioni, 1+49 e 25+25, mentre 8125 è il primo numero ad avere ben cinque diverse scomposizioni).
b. Generare le sequenza di tutti i numeri scomponibili come somma di due quadrati in due modi, in tre modi, in quattro modi… (sempre considerando numeri fino ad un valore massimo N)
c. Verificare anche la scomponibilità in somma di **tre** quadrati.

```
//Soluzione parziale


int main() {
  int N,a,b;
  scanf("%d",&N);
  for(a=0;a*a<N;a++){
     for(b=0;b<=a;b++){
        if(a*a+b*b==N)
           printf("a=%d,b=%d\n",a,b);
     }
  }
  return 0;
}
```

Sottosequenze di numeri ordinati

Scrivere un programma che legge da stdin una sequenza (di lunghezza arbitraria) di numeri interi positivi, terminata da 0, e indica, alla fine delal sequenza, qual è la lunghezza della massima sottosequenza di numeri consecutivi in ordine crescente. Esempi:

```
13   3   8   4   5     1  17   0
Lung. max = 2
```

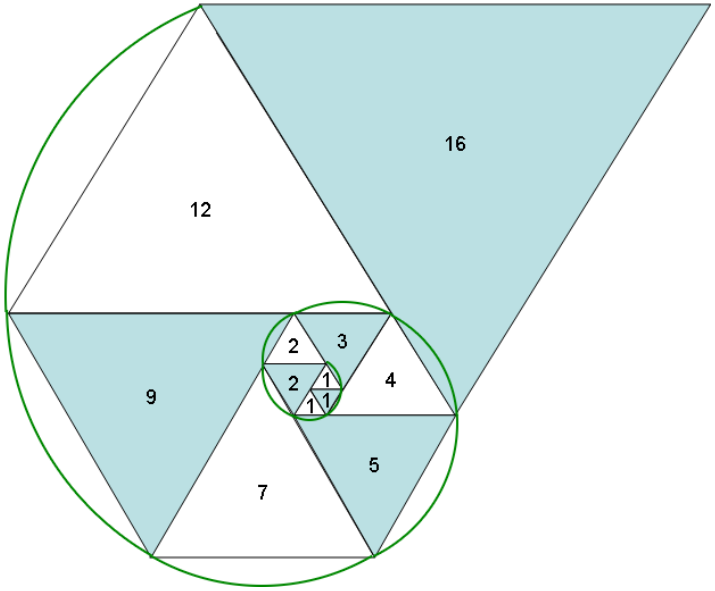
```
21   19   18   14   9   6   4   3  0
Lung. max = 1
```

```
2   1   3   6   8   5   1  12  18  17   0
Lung. max = 4
```

```
#include <stdio.h>
int main() {
  int val,cont=0,max=0,val2=0;
  do{
    printf("dammi num\n");
    scanf("%d",&val);
    if(val!=0){
      if(val>val2){
        cont++;
        if(cont>max)
           max=cont;
      } else {
         cont=1;
      }
    val2=val;
    }
  } while(val!=0);
  printf("max seq=%d\n",max);
  return 0;
}
```

```
#include<stdio.h>

int main(){
        int i=0,max=1,cont=1,prec,toDebug=0;

        printf("...\n"); scanf("%d",&prec);
        if(prec!=0) { printf("...\n");  scanf("%d",&i); }
        while(prec!=0 && i!=0){
                if(prec<i){
                        cont++;
                        if(cont>max)
                                max=cont;
                        if(toDebug)
                                printf("cont=%d\n",cont);
                } else {
                        cont=1;
                        if(toDebug)
                                printf("cont=%d\n",cont);
                }
                prec=i;
                printf("...\n");  scanf("%d",&i);
        }

        printf("max=%d",max);

        return 0;
}
```

Successione di Padovan



La successione di Padovan è la serie di numeri naturali P(n) definita dai valori iniziali:

$P(0) = P(1) = P(2) = 1$

E per tutti i valori di n > 3 dalla relazione:

$P(n) = P(n-2) + P(n-3)$

I primi valori della successione sono:

1, 1, 1, 2, 2, 3, 4, 5, 7, 9, 12, 16, 21, …

Scrivere un programma che chiede all'utente un numero intero e verifica se il numero inserito (che deve essere positivo) è uno degli elementi della successione di Padovan.

Suggerimenti:

- Verificare che il numero immesso dall'utente sia strettamente positivo e, se necessario, ripetere l'acquisizione del numero sino ad ottenerne uno valido
- Non è agevole fare una verifica diretta sul numero inserito dall'utente. Conviene piuttosto calcolare a (partire da 1,1,1) tutti gli elementi della successione, in ordine crescente, fino a quando è possibile verificare l'appartenenza o meno del numero inserito da terminale
- Si dichiarino e inizializzino opportune variabili dedicate a calcolare la sequenza di Padovan (tramite un ciclo che ad ogni iterazione calcoli l'elemento successivo)
- Si badi a gestire correttamente l'uscita dal ciclo

```
#include<stdio.h>

int main(){
        int val,pr=1,se=1,te=1,qu=1,toDebug=1;

        do{
           printf("int>0\n");
           scanf("%d",&val);
        }while(val<=0);

        while(qu<val){
           qu=pr+se;
           if(toDebug==1)
              printf("pr=%d se=%d te=%d qu=%d \n",pr,se,te,qu);
           pr=se;
           se=te;
           te=qu;
        }

        //qu>=val
        if(qu==val)
           printf("%d appartiene a Padovan \n",val);
        else
           printf("%d non appartiene a Padovan \n",val);
        return 0;
}
```

Esercizio

- Si realizzi un programma in linguaggio C per risolvere equazioni di secondo grado.
- In particolare, data una generica equazione di secondo grado nella forma
- $ax^2 + bx + c = 0$
- dove a, b, c sono coefficienti reali noti e x rappresenta l'incognita, il programma determini le due radici x_1 ed x_2 dell'equazione data, ove esse esistano.
- Si identifichino tutti i casi particolari (a = 0, b= 0, c=0) e si stampino gli opportuni messaggi informativi.

Esercizio

- Si realizzi un programma che legga da tastiera un valore intero N, compreso tra 1 e 10, e stampi a video un “quadrato di asterischi” di lato N.
- Esempio con N=5

```
*****
*****
*****
*****
*****
```

- Si realizzi una variante del programma per visualizzare solo i lati del quadrato
- Si realizzi una variante del programma per visualizzare un triangolo isoscele rettangolo di lato N
- Si realizzi una variante del programma per visualizzare solo i lati di un triangolo isoscele rettangolo di lato N

Esercizio

- Scrivere un programma in linguaggio C per la rappresentazione del triangolo di Floyd. Il triangolo di Floyd è un triangolo rettangolo che contiene numeri naturali, definito riempiendo le righe del triangolo con numeri consecutivi e partendo da 1 nell'angolo in alto a sinistra.
- Si consideri ad esempio il caso N=5. Il triangolo di Floyd è il seguente:

  1

  2 3

  4 5 6

  7 8 9 10

  11 12 13 14 15
- Il programma riceve da tastiera un numero intero N. Il programma visualizza le prime N righe del triangolo di Floyd.
- Suggerimento. Si osserva che il numero di valori in ogni riga corrisponde all'indice della riga: 1 valore sulla prima riga, 2 sulla seconda, 3 sulla terza.
- Estensione: si risolva il problema usando un ciclo solo

```
#include<stdio.h>
#define N 22
int main(){
        int i=1,j=1,k;
        while(i<=N){
                //ciclo da 1 a j
                for(k=1;k<=j && i<=N;k++){
                    printf("%d ",i);
                    i++;
                }
                j++;
                printf("\n");
        }
        return 0;
}
```